Improving Multi-Core Architecture Power Efficiency through EPI Throttling and Asymmetric Multiprocessing

Bob Crepps

Table of Contents

(Click on page number to jump to sections)

Improving Multi-Core Architecture Power Efficiency through EPI Throttling and Asymmetric Multiprocessing

Overview: The New Era of "Multi-Everywhere"

As the world looks to multi-core processors for new performance gains in mainstream computing, we're entering a new era of computing. You could call it the era of "multi-everywhere." This new era involves running multithreaded applications on multi-core processors that enable single-die multiprocessing. This new era is the next logical step in driving Moore's Law into another decade.

Yet this next step is not without challenges. One of them is Amdahl's Law. It says that the speedup possible through parallelism (the concurrent execution of tasks to achieve higher performance) will be limited by the portion of the time spent in the sequential component (or serial component) of an application. In other words, no matter how fast the multithreaded parts of a program run, the speedup of the overall program will be limited by the sequential portion of the computation.

This article focuses on a possible solution for improving the performance and energy efficiency of the sequential component in combination with the parallel components of program execution. The solution is EPI throttling. Using it, we can dynamically vary the energy per instruction (EPI) in different cores according to the amount of parallelism available in an application. The result is an asymmetric multiprocessing (AMP) solution that efficiently addresses parallel and sequential parts of the code. Using AMP with EPI throttling, Intel researchers have demonstrated a 38 percent wall clock (real-time) speedup for a wide range of multithreaded programs with sequential components.

Such dramatic results make a compelling case for EPI throttling as a solution for mitigating the effects of Amdahl's Law. They also point to EPI throttling as an important tool for another pressing issue: improving power efficiency in modern microarchitectures. This is another important challenge we face entering this new era of computing, and we will explore it here as well.

The Road Taken

To understand the forces behind this new era of computing, it helps to take a brief look at the path we've taken to improve processor efficiency.

Intel ushered in the personal computing revolution with its 8086, 8088, and later, 80236 processors. With the Intel386™ processor, Intel launched the era of instruction pipelining—enabling the processor to begin work on the next instruction before the previous one is complete. The next major innovation was the first fully pipelined Intel® architecture processor, the Intel486™ processor. This was capable of executing one instruction per clock cycle and featured another important first: an on-die cache. Speed was increased by using instruction pipelining to predict the next instructions and storing them in the cache, allowing the processor to pull data out of the cache as needed rather than using valuable overhead to access external memory. Intel then brought out the Intel® Pentium® processor and entered an era of superscalar architecture (the ability to execute two instructions per clock cycle) and branch prediction (a technique that attempts to infer the next instruction address to speed up performance).

The next big development was the Intel® Pentium® Pro processor. It introduced instruction-level parallelism to personal computing and was Intel's first processor capable of "out-of-order" processing—using multiple execution units operating on the same processor to enable instructions to be executed out of order.

The Intel® Pentium® 4 processor (with Intel NetBurst® microarchitecture) enabled even greater performance gains through innovations, enabling the handling of even more instructions per clock cycle and the ability to reach even higher frequencies. It significantly increased the number of pipeline stages to 20, and added an execution trace cache that stored decoded IA-32 instructions. This helped Intel keep the pipeline full and realize even greater performance.

Intel then inaugurated our current new era of computing with the introduction of Hyper-Threading Technology (HT Technology) in the Intel Pentium 4 processor. HT Technology enables a single processor to run two threads

simultaneously. With it, we entered the age of thread- and processor-level parallelism. But this is only the beginning.

In the next few years, this age of parallelism will be characterized by:

- Enabling ever greater amounts of multiprocessing through multi-core processors with ever greater numbers of cores
- Developing greater amounts of parallelism in software code to more fully utilize processor resources for higher performance
- Bringing high-performance computing applications (such as those used for recognition, mining, and synthesis) into mainstream computing to handle the enormous amounts of data the world now generates daily

The Need for Greater Power Efficiency

Throughout this journey from pipelined to multi-core architecture, Intel has continually extended and expanded Moore's Law by shrinking the already minuscule size of silicon devices. Today Intel researchers face new challenges: working within the physical limits of atomic structure for scaling transistors while managing both power and heat. The problem is that while smaller transistors consume less power, the overall chip consumes more power and generates more heat as transistor density and speed increase. (Moore himself raised the prospect of this thermal problem in his original paper by saying, "Will it be possible to remove the heat generated by tens of thousands of components in a single silicon chip?")

In response to these growing power and thermal challenges, Intel is aggressively pursuing research into both conventional and unconventional technologies. This research includes introducing many new and exciting technologies and innovations in materials, design and architecture. One promising direction is power reduction techniques. The power reduction techniques we will deal with in this article include multi-core processors, multithreading, chip multiprocessing, and EPI throttling.

Power Reduction and the New Definition of Performance

The need for power efficiency is driving a new definition for performance. Performance is now a factor of speed plus power efficiency. The way we primarily achieve this new definition of performance is through multi-core processors (two or more full CPU cores within a single processor).

For complex workloads having data and task parallelism that can be accelerated by multithreading and multiprocessing, there is a nearly linear speedup with increasing cores. (See Figure 1.)

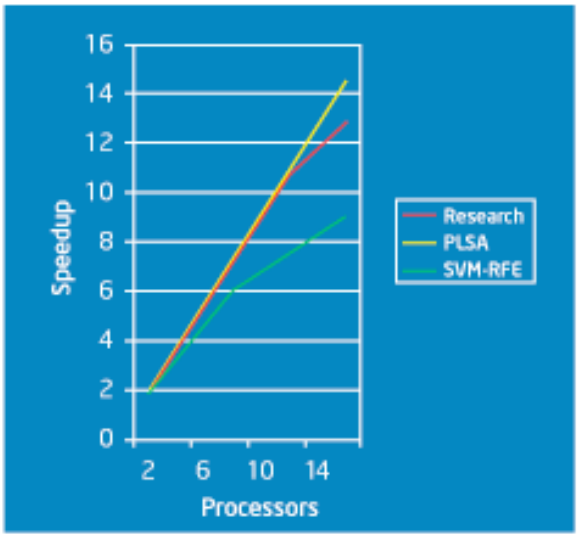

Figure 1. Common benchmarks for complex workloads show a nearly linear speedup with increasing cores.

Of course, challenges exist today in creating the necessary parallelism in applications to see these kinds of performance gains. But these challenges are being worked on. (For one promising technique for parallelizing some of the applications most resistant to parallelization, see the white paper “Mitigating Amdahl’s Law through EPI Throttling”.)

Such work is important because simultaneous multithreading (parallel processing) is particularly power-efficient when run through multi-core processors. This is because it is more power efficient to have multiple small cores each run individual threads than to have a single large processor run multiple threads. A multi-core design also enables cores to share processor resources such as cache. The resulting efficiencies allow multi-core designs to achieve an increase in simultaneous multithreading performance without a corresponding increase in power (see Figure 2).

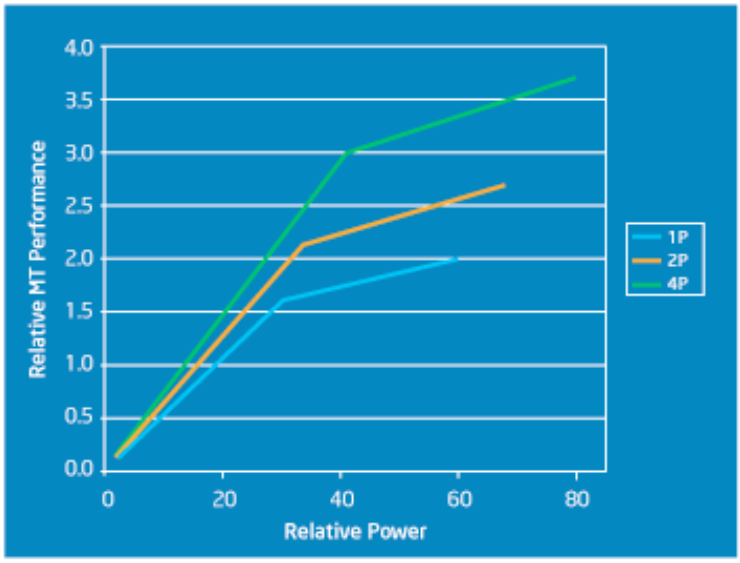

Figure 2. Relative multithreading (MT) processing performance increases with the number of cores while achieving greater power efficiency.

Multithreading also improves power efficiency by maintaining a higher level of processor utilization. This is important because in a system designed for full-power thermal and power delivery, the busier the processor, the greater the efficiency. Fixed power costs (clock distribution and leakage) are amortized over more work done. If only a single thread is running, techniques such as speculative and out-of-order execution can only go so far in maximizing utilization. When main memory access is required, processing stalls and power is wasted. Multithreading is much different. Since there are more threads to work with, when the first thread stalls waiting for memory, a second thread launches. When it stalls, a third thread launches. In this way, multithreading enables a processor to remain fully utilized, achieving higher performance within the same amount of power.

These performance gains become more pronounced when you combine multithreading and multiprocessing with a multi-core processor. Figure 3 shows that the cost of a unit of performance for a single-core processor is more than one unit of power and die area.

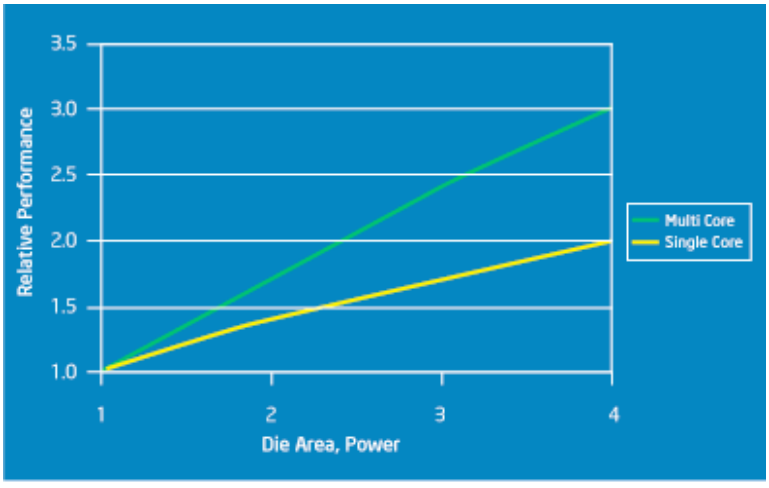

Figure 3. Running multithreaded applications on multi-core processors enables higher performance in the same power envelope.

It also shows that for multi-core processors, the same increase in die area and power achieves a nearly linear increase in

performance. This is because the cores in a multi-core processor share cache and the frontside bus, providing more computing power without having to duplicate all of the other logic in a single-core processor. Adding additional multithreading capabilities to each core through HT Technology enables even higher performance in the same power/die-area envelope.

Reality Check: Amdahl's Law

Achieving such performance gains in the same power envelope is contingent upon applications that are ideally suited for simultaneous multithreading. But what about Amdahl's Law? It would caution that parallel speedups like these are limited by the amount of code that is scalar (sequential and unable to be threaded). In other words, the parts of an application that cannot be multithreaded will determine the speed of the processing.

Consider, too, that some programs have well-defined parallel and sequential phases, whereas in others the sequential portion is a moving target. Imagine a multithreaded program in which multiple threads normally run in parallel, but occasionally synchronize (wait) on each other. The last thread running while all the others are waiting becomes the sequential portion. And the limiting factor.

Figure 4 shows just how big an effect Amdahl's Law can have on performance as an increasing amount of code is scalar. The top result (s = 0) shows a linear speedup of early 100 with an increasing number of cores. The middle result (s = 0.1) shows how much performance drops when just 10 percent of the code is scalar. The speedup is reduced to just 10. The bottom result (s = 0.5) shows how the speedup drops to just 2 when the scalar code is 50 percent.

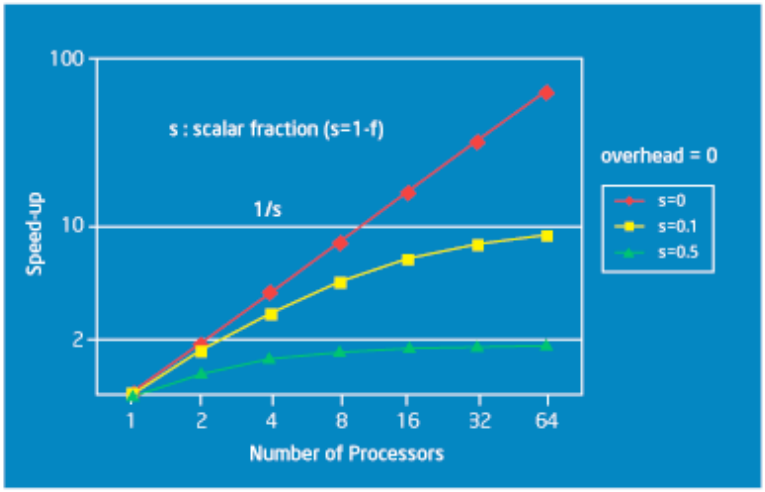

Figure 4. The impact of scalar code on multi-processing over multiple cores.

How can we overcome these performance hits from scalar code?

Asymmetric Multiprocessing (AMP) and EPI Throttling

Obviously, because of Amdahl's Law, the first challenge for multiprocessing performance is to improve the performance of the scalar components of execution. The second challenge is to do this within the constraints of a power budget, since power/thermal issues are now an increasingly critical design and performance constraint. The third challenge is to satisfy the conflicting microarchitectural demands of parallel and sequential processing.

It is a seemingly impossible task to optimize a processor for both single-threaded (sequential) and throughput (parallel or multithreaded) performance within a fixed power budget. Most of today's microarchitectural techniques (such as out-of-order execution, speculation, deep pipelining, and so on) help reduce latency for single-threaded execution, but do it at a relatively high EPI. This high EPI limits the number of CPU cores that can be accommodated in a multi-core processor for a given power budget. Conversely, multiprocessing throughput performance demands exploiting thread-level parallelism with a multi-core processor having many low-energy cores. How can you have both?

To find out, Intel researchers experimented with two factors. The first was EPI throttling—dynamically varying the EPI according to the amount of parallelism available in the software. The second factor was AMP. To implement EPI throttling and set up a prototype of an asymmetric multiprocessor with four cores, they used an off-the-shelf Intel® Xeon® processor–based 4-way symmetrical multiprocessing (SMP) server. Through a novel combination of the Intel Pentium 4 processor's clock throttle mechanism, along with the ability of Linux* to assign processor affinity (binding a process or a set of processes to a specific processor or a set of processors), the researchers created multiple performance and power operating points. These points enabled them to simulate a multi-core processor where each "core" expends a varying amount of energy per retired instruction, by varying individual processor voltage and frequency based on the available thread-level parallelism (see Figure 5). (Note: The AMP prototype didn't actually alter voltage or frequency. A clock throttle enabled the researchers to measure the performance effects of a CPU that runs at a specified frequency.)

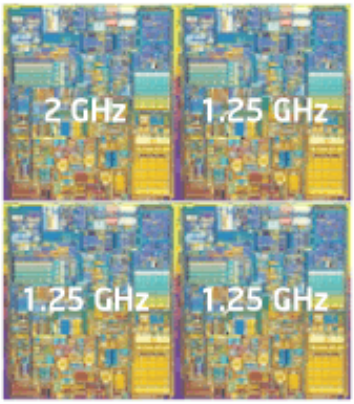

Figure 5. In an asymmetric multiprocessor, the cores can run at different speeds to dynamically vary the EPI.

With this asymmetric multiprocessor prototype, parallel phases of code were dynamically assigned to the low EPI processors in the AMP prototype, and sequential phases were assigned to the high EPI processor. This meant that if there were four available threads in a given phase, they were processed by all four cores running at 1 GHz. If the number of threads dropped to two, the power budget was reassigned to just two processors running at 1.5 GHz. During a sequential phase, the power budget was assigned to a single processor running at 2 GHz. In this way, the power consumed by the asymmetric multiprocessor was constant all through the program run. The EPI per core was lower during parallel execution phases. In sequential execution phases, the EPI for the single active core was higher to increase the speed of processing sequential execution phases.

Comparing the Performance of SMP and AMP

To evaluate the performance of AMP using EPI throttling, Intel researchers created for comparison a prototype of a symmetrical multiprocessor using the exact same "cores." This symmetrical multiprocessor used the same power budget, with each core running at the same speed (1 GHz) at all times. Both AMP and SMP prototypes were given a power budget equivalent to the power consumed by a single Intel Xeon processor running at 2 GHz.

The researchers ran a range of realistic, multithreaded programs, such as bioinformatics programs, decision-support programs, and a parallel Fourier-transform solver. These programs provided different ratios of sequential-to-parallel execution. For instance, in one bioinformatics program, 30 percent of the execution time was spent on a sequential operation while the remaining 70 percent was spent on parallel operations.

The results fell in three categories (see Figure 6).

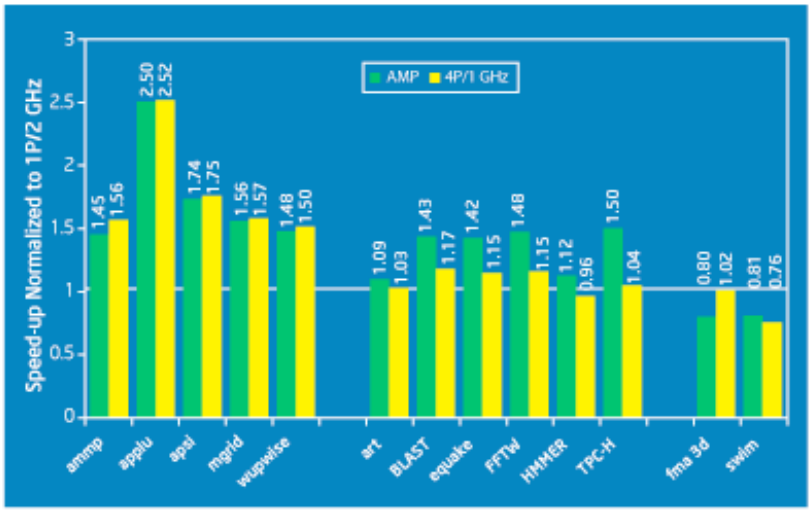

Figure 6. AMP provides a big performance boost on moderately parallel programs (center group).

- For highly parallel programs (grouped on the left side of the graph), both AMP and SMP performed equally well, providing real performance gains.
- For moderately parallel programs (grouped in the center of the graph), AMP averaged a 38 percent wall clock speedup over SMP. These programs spent about 23 percent to 36 percent of their execution time in sequential processing.
- For highly sequential programs (grouped on the right side of the graph), both AMP and SMP provide no performance benefit over a standard 2 GHz processor.

An explanation for why AMP improves performance can be found in Figure 7.

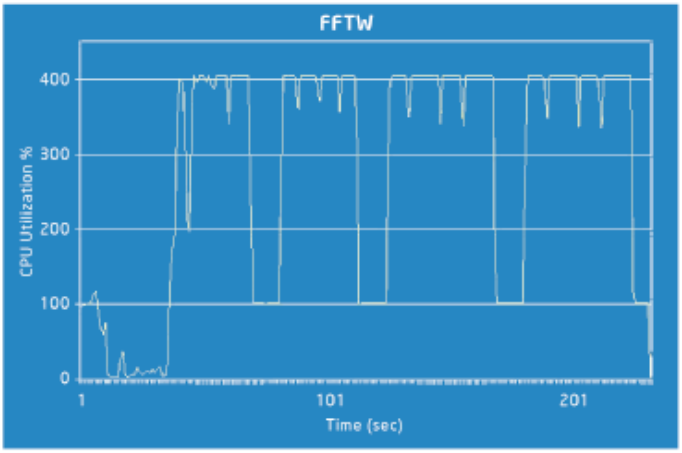

Figure 7. AMP results running an application with four distinct transitions between sequential and parallel phases.

This graph shows an AMP run of FFTW (a collection of programs for computing the Discrete Fourier Transform in multiple dimensions). The graph shows distinct phases of sequential and parallel execution where the processor utilization is 100 percent (one processor) and 400 percent (all four processors) respectively. AMP uses a single processor running at 2 GHz during sequential phases and uses four processors running at 1 GHz during parallel phases. The graph shows that AMP could improve performance by varying the EPI according to the number of threads to continuously optimize the use of a given power budget.

In contrast, using SMP with four processors running at 1 GHz SMP on the same application would underutilize the power budget during sequential phases when only one processor is running. And a single processor running at 2 GHz would not take advantage of available thread-level parallelism during parallel phases.

The best representation of the value of AMP and EPI throttling can be found in Figure 8, which graphs all the results against an increasing sequential component. The lower the run time (y axis), the better the performance. While SMP loses its performance advantages to a single processor running at 2 GHz when the sequential components near 35 percent, AMP shows performance improvement over a single processor even when the code is nearly 100 percent sequential.

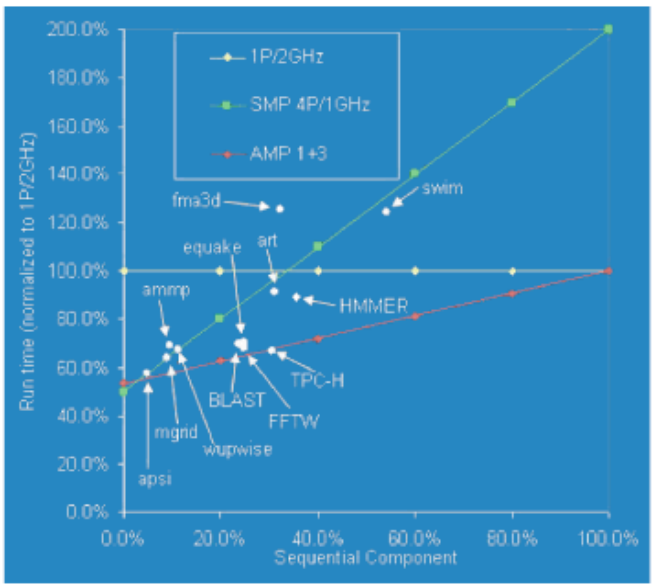

Figure 8. A comparison of AMP vs. SMP performance as the sequential component increases

Summary

The current shift toward a new era of computing based on multiprocessing, multithreaded applications on multi-core processors comes at a time when processor power/thermal properties are increasingly challenging to control. Moving forward requires researchers to find new ways of achieving performance gains through better design while staying within a given power budget.

Intel researchers believe EPI throttling will be an essential aspect of future multi-core processors, monitoring the activity levels of the various cores and, based on that activity, determining when to reassign tasks to particular cores and when to reduce power consumption. Actual implementations of an EPI throttle will probably be as a hardware mechanism that operates transparently to software. Enhanced Intel SpeedStep® Technology on the Intel® Core™ Duo and Intel Core Solo processors already uses EPI throttling to adjust the voltage and frequency operating point in response to a thermal sensor or software control.

Tests have shown that an EPI-throttled asymmetric multiprocessor can dramatically improve performance and significantly mitigate the effects of Amdahl's Law while running substantial amounts of both sequential and parallel code. AMP and EPI throttling together represent a novel way to take advantage of a variety of today's technologies (multi-core processors, multithreading technology, and flexible power-down techniques) to dramatically improve performance within a fixed power budget.

More Info

You can learn more by reading the following:

- "Mitigating Amdahl's Law through EPI Throttling"(white paper)
- "Fixing the Sequential Bottleneck by Regulating Energy per Instruction on CMPs"(Technology@Intel Magazine article)

—End of Technology@Intel Magazine Article—